PCT/GB 2005/000231

PATENTS • DESIGNS
The Patent Office
COPYRIGHT • TRADE MARKS

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed [illegible]

Dated 9 February 2005

25-MAR-2004 16:38 FROM MARKS & CLERK TO PAT OFF GB P.03

Patents Form 1/77

Patents Act 1977
(Rule 16)

THE PATENT OFFICE
DL
25 MAR 2004
RECEIVED BY FAX

The Patent Office

25MAR04 E883926-1 D10176
P01/7700 0.00-0406722.9 ACCOUNT CHA

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

25 MAR 2004

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference	GBP290155
2. Patent application number (The Patent Office will fill in this part)	0406722.9
3. Full name, address and postcode of the or of each applicant (underline all surnames)	Cryptomathic A/S, Kannikegade 14, 3 Aarhus C DK-8000 Denmark
	6711661002
Patents ADP number (if you know it)	
If the applicant is a corporate body, give the country/state of its incorporation	Denmark
4. Title of the invention	A Voting System With Full Accountability
5. Name of your agent (if you have one) "Address for service" in the United Kingdom to which all correspondence should be sent (including the postcode)	Marks & Clerk 66-68 Hills Road Cambridge CB2 1LA
	7271125003
Patents ADP number (if you know it)	~~18001~~

6. Priority: Complete this section if you are declaring priority from one or more earlier patent applications, filed in the last 12 months	Country	Priority application No (if you know it)	Date of filing (day / month / year)

7. Divisionals, etc: Complete this section only if this application is a divisional, application or resulted from an entitlement dispute	Number of earlier application	Date of filing (day / month / year)

8. Is a Patents Form 7/77 (Statement of inventorship and of right to grant of a patent) required in support of this request? Yes
(Answer 'Yes' if:
a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

0098903 25-Mar-04 03:45

Patents Form 1/77

9. Accompanying documents: A patent application must include a description of the invention. Not counting duplicates, please enter the number of pages of each item accompanying this form:

Continuation sheets of this form	0
Description	33 /
Claim(s)	2
Abstract	
Drawing(s)	

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent (Patents Form 7/77)	
Request for preliminary examination and search (Patents Form 9/77)	
Request for substantive examination (Patents Form 10/77)	
Any other documents (please specify)	

11. I/We request the grant of a patent on the basis of this application.

Signature(s) Marks & Clerk Date: 25 March 2004

12. Name and daytime telephone number of person to contact in the United Kingdom

Cambridge Office
01223 345520

A Voting System with full Accountability

Gorm Salomonsen, Jens Groth, March 4 2003.

Introduction

An election poses a lot of challenges on the system used for voting, whether this is a manual system, a mechanical one or an electronic one. Traditionally manual systems have been used and are still widely used. For some decades mechanical systems have been used in some countries, and in recent years electronic voting systems have had their breakthrough in a number of countries. Common to all is that very high standards have to be set on the security of the process of voting; such that voters can be confident that the result of the election correctly reflects the votes cast, whereas at the same time secrecy of the votes cast shall be ensured. In fact a long list of apparently conflicting requirements can be stated.

Common for the systems used for general elections in a larger scale today is that they duplicate the basic principles of the manual election, which we will briefly review. A voter enters a voting site, where his identity is checked, after which he receives a ballot and enters a voting booth where he can vote in privacy. He then folds his ballot such that nobody can see what he has voted, enters the public sphere again and drops his ballot into a container. The whole process is monitored by a sufficiently large and diverse group of people such that it can be trusted not to cheat. A number of special cases may exist in the process. For example the first voter may have the opportunity to verify that the container is initially empty and it may be possible to regret the choice in the time span between entering the choice on the ballot and dropping it into the container. After the election the votes are counted. Throughout the whole process it is ensured that at every step everything is monitored by a group of people sufficiently large and diverse to be trusted.

Mechanical and electronic voting systems follow the same principles. In fact it seems that the core element in the design of such systems is that the process shall be changed as little as possible when introducing a new system. For example DRE (Direct Recording Engine) e-voting systems, store individual votes on a memory card such that they can be counted afterwards instead of just keeping track on the statistics to be reported.

However, when using electronic devices a number of properties of the original process are altered in disfavour of the security despite that the process is kept fixed. In particular the following properties are always lacking unless great care is taken:

a) The voter is no longer able to see that what he enters on the machine is actually what is recorded.
b) The officials monitoring the process are no longer able to see that one vote is recorded for each voter.
c) The monitoring of the counting process is no longer efficient since nobody can see what really happens during counting.

This has been known for many years in academic circles and has led to a number of initiatives:

1) Some have tried to inform the public and decision makers about the situation and have been driving a debate that has recently been rather heated as DRE machines have become more widespread.
2) Some have developed the technology for dealing with the new challenges posed by electronic voting system. This has been done as basic research in universities worldwide and in applied research projects like the e-Vote (IST 2000-29518, http://www.instore.gr/evote) and Cybervote (IST-1999-20338, http://www.eucybervote.org) projects as well as in private high tech companies like Cryptomathic.

For background prior art references can be made to the following:

[DGS] Ivan Damgård, Jens Groth, Gorm Salomonsen "The Theory and Implementation of an Electronic Voting System", In Gritzalis, D. (Ed.) Secure Electronic Voting, Kluwer Academic Publishers, Boston, USA, November 2002 (ISBN 1-4020-7301-1)

[DJ01] Ivan Damgård and Mads Jurik. "A generalisation, a simplification and some applications of Pailliers public-key system with applications to electronic voting", In Public Key Cryptography '01, pages 119-136. Springer-Verlag, LNCS 1992, 2001.

[NEF01] C. Andrew Neff. "A verifiable secret shuffle and its applications to e-voting". In proceedings of the 8'th ACM conference on Computer and Communications Security, pages 116-125. ACM Press, 2001.

[NEF03] C. Andrew Neff "Election Confidence". Version 6, December 2003. Preprint available on www.votehere.net.

[BGR] Mihir Bellare, Juan A. Garay and Tal Rabin: "Fast Batch Verification for Modular Exponentiation and Digital Signatures", EUROCRYPT 1998, LNCS series 1403, Springer Verlag, pages 236-250.

[DF] Ivan Damgård and Eiichiro Fujisaki: "A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order", ASIACRYPT 2002, LNCS series 2501, Springer Verlag, pages 125-142

[F] Jun Furukawa: "Efficient, Verifiable Shuffle Decryption and Its Requirement of Unlinkability", Public Key Cryptography 2004, LNCS series, Springer Verlag, pages 319-332.

[FMMOS] Jun Furukawa, Hiroshi Miyauchi, Kengo Mori, Satoshi Obana and Kazue Sako: "An Implementation of a Universally Verifiable Electronic Voting Scheme based on Shuffling", Financial Cryptography 2002, LNCS series 2357, Springer Verlag, pages 16-30.

[FS] Furukawa and Sako: "An efficient scheme for proving a shuffle", CRYPTO 2001, LNCS series 2139, Springer Verlag, pages 368-387.

[G] Jens Groth: "A Verifiable Secret Shuffle of Homomorphic Encryptions", Public Key Cryptography 2003, LNCS series 2567, Springer Verlag, pages 145-160

[GMY] Juan A. Garay, Philip D. MacKenzie and Ke Yang: "Strengthening Zero-Knowledge Protocols Using Signatures", EUROCRYPT 2003, LNCS series 2656, Springer Verlag, pages 177-194.

[Npatent] Andrew Neff, VoteHere: "Verifiable secret shuffles of encrypted data, such as ElGamal encrypted data for secure multi-authority elections", patent application 2002.

Pursuers of 1) require printed ballots to be produced for voters to watch and store the traditional way such that they can be used for recounting. The pilot system developed and tested in the e-Vote project uses digitally signed, encrypted votes, such that it is ensured that there is control of, who cast each individual vote. It also utilizes a secure protocol based on homomorphic encryption and zero-knowledge proofs (see [DGS], [DJ01]) to ensure that the counting process is universally verifiable while preserving secrecy. *Universally verifiable* means that it is possible for an independent observer to verify that the votes are authentic, correctly formatted and have been counted correctly without breaking the secrecy of the election. However, it does not deal directly with the issue mentioned in a), that each voter shall be able to verify that his choice is actually what is recorded in his vote[1].

One purpose of embodiments of the present system is to bring together the two approaches in a novel way by outlining how an e-voting system can be designed with existing technology such that

I. The properties of embodiments of the system are such that none of the issues a), b) or c) constitutes a significant security treat.
II. Several counting and recounting procedures are possible with different properties with respect to security and cost and where the highest obtainable level of integrity of the result of the election is considerably higher than for traditional manual elections.

Thus in a relaxed political climate costs can be saved and final results of the election can be made available quickly, whereas in a tense political climate, where current manual procedures are insufficient to ensure integrity of elections, the level of security can be increased.

[1] Votes with the e-Vote system are generated and signed in an applet on the PC of the voter, so a) can be ensured by intercepting the applet and verifying that it performs correctly (by means of installing third-party software). However, this only works for Internet voting and it comes together with the expense that receipt-freeness is only conditionally possible with Internet voting.

What is claimed is:

1. A voting system feature comprising: at least one device used for voting entering preferably (the same or associated information) on a printed ballot and an encrypted electronic ballot linking the two to each other; preferably. Each voter will be allowed to watch the content of the paper ballot to verify that it contains his choices preferably. At least one instance making available depersonalised clear-text electronic ballots with their information linking them to printed ballots to the public or to selected entities; preferably. A procedure selecting a random sample of electronic ballots and verifying that their content correspond to the content of corresponding paper ballots with the purpose of establishing confidence that the electronic ballots have not been subjected to large-scale tampering.

2. A voting system feature comprising: at least one device used for voting entering preferably (the same or associated information) on a printed ballot and an encrypted electronic ballot linking the two to each other; preferably. Each voter will be allowed to watch the content of the paper ballot to verify that it contains his choices preferably. At least one instance making available depersonalised clear-text electronic ballots with their information linking them to printed ballots to the public or to selected entities; preferably. A procedure selecting a random sample of electronic ballots and verifying that their content correspond to the content of corresponding paper ballots with the purpose of establishing a deterrent against tampering with the voting device in individual election districts.

3. A device for collecting ballots comprising: two or more containers for collecting filled ballots and a user interface allowing a voter to make aware of his intention to submit his ballot arranged in such a way that it is decided at random at the time of ballot submission whether ballots shall be checked. This works in the way that it is by mechanical means ensured that ballots selected for checking at random are entered in a particular subset of containers.

4. A protocol for producing a zero-knowledge proof of a correctly performed combination of permuting and partial decryption of homomorphically encrypted messages and preferably the non-interactive versions of the protocol obtained by using the Fiat-Shamir heuristic.

5. A homomorphic commitment system that performs efficiently by making use of subgroups of Zn* for the message space and/or the randomization space.

6. A protocol comprising: use of a homomorphic verification system for verifying the correctness of the result of repeatedly permuting and re-encrypting and finally decrypting homomorphically encrypted content.

7. A protocol comprising: use of a homomorphic verification system for verifying the correctness of the write-in votes obtained by repeatedly permuting and re-encrypting and finally decrypting homomorphically encrypted votes.

8. A protocol comprising: use of a homomorphic verification system for verifying the correctness of the information linking electronic and printed ballots obtained by

repeatedly permuting and re-encrypting and finally decrypting homomorphically encrypted votes.

Aspects of the invention provide data processing apparatus and computer program code (which may be distributed over a network), in particular on a carrier, to implement the above described system and protocols.

We are thus offering a new solution that allows for faster counting, cost savings and increased service to voters compared to manual elections, but with a higher level of security. We must stress that aspects of the invention can be used in many embodiments. There are many technologies available for dealing with the issues a) and b) and many possible embodiments which we will not exhaustively list in this document. In particular all of the technologies "homomorphic encryption", "MIX nets" and "digital signatures" can be replaced by other technologies in the embodiments without changing the use of claim 1, 2 and 3.

Overview

When we discuss technologies suitable for protecting elections it will be technologies that base their trust on mathematics and suitably composed groups of people being unable to cooperate to cheat rather than in elements like trust in the quality of code or ability to keep out intruders completely. For example a digital signature cannot be forged by malicious software that has access to data that can be signed unless this software also has access to a particular private key. This is contrary to other sorts of protection, like a log on a local machine that can normally easily be forged by malicious software. Thus the protection we discuss is protection against adversaries with access to modifying any part of the software they like with very few exceptions (software for key generation is an example). When we state that a device must be trusted to do or not to do something, we mean that we rely on that the software and hardware of the device ensures that the device has the intended behaviour. The precise level of security for the devices used for casting votes, we are aiming at, is:

- The devices will be trusted not to give away the choices of individual voters in any other ways than the ones specified.
- However, we will assume that relevant adversaries have access to modifying the software and hardware of the devices whenever we discuss the highest levels of security supported for protecting against tampering with the choices of the voters.

This is consistent with the fact that the latter type of attack has the highest potential for producing benefits for adversaries, and with that also manual voting allows some attacks, like the use of hidden cameras or comparison of fingerprints on voter cards and ballots, for breaking the secrecy.

Two technologies for counting secret encrypted and signed votes (the list is not exhaustive, the ones mentioned are the ones we are particularly interested in making use of in our invention) are:

- Homomorphic encryption and zero-knowledge proofs combined with a secret sharing mechanism. The vote is encrypted and a zero-knowledge proof is attached proving that the encrypted vote is an encryption of a correct vote. Because the crypto system is homomorphic the votes with correct zero-knowledge proofs can be counted on encrypted form without ever decrypting a vote. Finally, the key for decrypting the result is secret shared between a sufficiently large and diverse group of people such that it can be trusted not to decrypt individual votes.

- MIX nets. A number of servers (shuffles) one after another re-encrypts encrypted votes without being able to decrypt them and passes them on in a different, random order together with a zero-knowledge proof that only the order but not the content of the encrypted votes has been modified. If several shuffles are used one after another and are operated by different organisations with conflicting interests, it is trusted that the association between the original ordering of the votes and the new ordering of differently encrypted votes has been lost. Further, the zero-knowledge proofs ensure that the content of the votes has not been altered. Again a secret sharing mechanism can be used for decryption.

Common to the two approaches is that they require the use of sophisticated zero-knowledge proofs and that crypto systems need to satisfy special properties in order to be used for the protocols. Until recently protocols of this type were too slow to be applied in practice, but currently:

- Cryptomathic has developed an efficient homomorphic encryption protocol and an efficient MIX net protocol. Both can be implemented over the same homomorphic crypto system.

We notice that the two technologies have different properties:

- Counting including verification can be parallelised arbitrarily for homomorphic encryption, so it scales well and can produce a fast result. Further it is easy to trace back votes that are incorrectly formatted electronic votes to their origin with this technology (this should never happen unless machines used for voting are malfunctioning or tampered with – instead there should be a correctly formatted invalid choice). The disadvantage is that a special zero-knowledge proof must be designed for each voting rule.
- MIX nets are more flexible when it comes to implementing different voting protocols because the same zero-knowledge proofs can be used for all voting rules.

In one of the proposed embodiments of our invention we will combine both technologies in order to get the best properties from both technologies.

The technologies discussed are sufficient to deal with the issues b) and c) mentioned in the introduction, so it remains to discuss the issue a). By having ballots printed voters are provided with the service that they can see what they have voted on paper, and they have the same level of certainty as at a manual election, that their vote will count, *provided that a manual recount actually takes place*. The idea, as already hinted, however has a number of shortcomings in its pure form:

- Almost no information is gained by checking a few votes in a district. The only action that makes sense is to make total recounts in a selection of districts.
- However, if let's say a manual recount takes place in 10% of the districts, this gives a 10% chance of being taken for somebody manipulating votes in a particular district for a particular election. This may well be a chance worth taking for a politician facing a ruined carrier if he looses. The same can be said for a 30% or a 50% chance.

Consequently quite comprehensive recounting is necessary in order to ensure that the mechanism works as intended – not only by revealing attempted fraud, but also by preventing attempts of fraud from happening by acting as a deterrent. Embodiments of an aspect of our invention have the following core properties:

- Electronic votes contain encrypted information identifying the manual vote and preferably the election district.
- The electronic votes can be detached from the identity of the voter by means of a MIX net or a similar mechanism in a secure way. After being detached from the identities of the voters, they are decrypted.
- We can pick a random sample of all the electronic votes of an arbitrary size.

Say that we want to ensure with 99% probability that at most 1% of the electronic votes are tampered with, i.e. contain different choices than the ones entered by the voters. Then we pick 459 random electronic votes. For each of those, if at least 1% of the electronic votes contain different choices than the corresponding manual votes, it has less than a 99% chance of passing the test of being compared to the corresponding manual vote. Consequently there is a probability of less than $0.99^{459} = 0.009921$ that all of them pass the test.

It is clear that letting electronic ballots identify non-existing printed ballots will be discovered. However, letting more electronic ballots identify the same printed ballot is a possible attack unless care is taken. The procedure that must be carried out in the individual districts is therefore *to run through all printed ballots in the district to establish that there is exactly one printed ballot with the same identification as the electronic ballot and that the choices on the printed ballot are the same as on the electronic one.*

For the ultimate case, a general election in the US say, it means that by manipulating 459 votes out of maybe 100.000.000 or even 200.000.000 and causing the rather simple procedure to happen in 459 randomly chosen election districts, you actually get quite confident that no large scale fraud takes place with the electronic votes. And this is by carrying out a procedure simpler than counting manually in less than 10 election districts in each state in average.

CLAIM: Let each encrypted vote carry information linking it to an individual ballot. After detaching the votes from the identities of the voters, take a sample of random decrypted electronic votes and compare them to the corresponding manual votes in order to create confidence in the accuracy of the result of the election with relatively little effort.

Comment: Additional information on an electronic ballot can be used for coercion by entities with access to decrypted, depersonalised electronic ballots. Therefore the information should be represented on the printed ballot in a form difficult to manage by voters (not easier to copy than taking a photo of the ballot or essential parts of the ballot) and the voter should preferably not be able to influence the information. One possibility is to use random data represented as bar-codes on the printed ballots.

Comment: The way statistics behave when doing different kinds of checking follows from elementary mathematics. The low efficiency of the standard scheme of producing manual ballots without any other option than doing full recounts for election sites or election districts was also noticed in [NEF03]. However, in [NEF03] using printed ballots was seen as opposed to using testing based on providing voters with receipts. In particular it is clear that the solutions covered do not have the novel property that efficient testing can be done without providing voters with receipts, which they may have difficulties with handling and understanding. Instead with embodiments of our invention voters are provided with a printed ballot, from which they can see directly what they voted. Consequently the usability properties of embodiments of our invention are superior compared to systems providing voters with receipts.

This scheme can also be carried out the other way around, in that paper ballots are picked and compared to anonymised electronic ballots. This has the advantage that less manual work is required. We propose the following scheme: the paper ballots are counted, the number is compared to the number of electronic ballots from the district. Then some paper ballots are picked at random and it is verified that they correspond to electronic ballots and have the same content. If the number of paper ballots and electronic ballots are not the same, the paper ballots are counted. The property we are aiming at using is that if there is the same number of electronic and paper ballots, and a certain number of electronic ballots do not correspond to paper ballots, then the same number of paper ballots do not correspond to electronic ballots. Thus, *if we know that all the paper ballots are different and the number of paper ballots correspond to the number of electronic ballots*, it is just as efficient to pick random paper ballots.

The procedure described above is efficient for revealing large-scale fraud. However, it still suffers from the deficit that it does not efficiently act as a deterrent against fraud in individual districts. Before we proceed with describing how to install such a deterrent, we will notice the difference between the requirement for having confidence in the overall accuracy of a country-wide election and the requirement for having a deterrent. The first needs to be established quickly such that the result of the election can take effect. For the latter to work, it is however enough that fraud is detected with a high probability inside a reasonable time window, for example a few months. That means that costs can be kept down when repeating the procedure in individual districts by having few MIX nets (and corresponding high-security facilities and staff) doing the electronic parts, and by giving districts reasonable deadlines for answering results such that they can organise their work efficiently. It also has the advantage that the capability of decrypting votes does not have to be distributed on too many facilities and persons.

We give an example of how an embodiment of another claim of our invention can act as an efficient deterrent.

Say we carry out the procedure described above with 194 randomly chosen votes in each district. Then in each district somebody manipulating 2% of the votes will face a 98% chance[2] that the fraud is detected. If he manipulates 1% of the votes he will face an 86% chance that it is detected and if he manipulates 0.5% of the votes he will face a 62% chance that it is detected. If he manipulates 0.1% of the votes, he will face a 17% chance that it is detected, which is not much, but on the other hand his chances of influencing the outcome of the election by changing 0.1% of the votes are probably also not good. If fraud is detected in this way, a manual recount and a police investigation can be initiated such that the result of the election can be corrected and such that apparently fraudulent candidates and their assistants can be tried in court.

The number of votes checked and the procedure that takes place in case that fraud is detected can of course be tuned according to needs.

CLAIM: Use information in each encrypted vote linking it to an individual ballot. After detaching the votes from the identities of the voters, take a sample of random decrypted electronic votes and compare them to the corresponding manual votes in order to install an effective deterrent against fraud with the election.

We must expect that both the procedure for creating confidence in the result of the election and the deterrent will be used together. Further, this will be done in a manner as efficient as possible. We describe a procedure below:

- At each election site/district there is a PC with a scanner capable of reading the information on the paper ballots linking them to electronic ballots, but not necessarily capable of reading what is voted for. The PC is on-line, is running a special application and has access to the electronic anonymised votes.
- The paper ballots are scanned and a program on the PC verifies that all the ballots carry different information, that the information corresponds to information on an electronic vote and that the number of paper votes is the same as the number of electronic votes.
- A sample of (about 194) randomly chosen votes is collected. For each of those it is verified that the electronic vote corresponds to the paper vote.

Public Key Cryptosystems

A public key cryptosystem consists of three algorithms K, E, and D.

- K is the key generation algorithm and produces a public key, pk, and a secret key, sk.
- E is the encryption algorithm. It takes as input the public key pk and a message m. It produces a ciphertext $c = E_{pk}(m)$.
  The algorithm may be randomized; it generates some random bits and uses

[2] Probabilities are estimated under the assumption that there are much more than 194 votes. Lower number of votes in all cases give higher probability of detection.

them in the encryption process. When emphasizing these random bits, we write them as an explicit extra input to the encryption algorithm, i.e., $c = E_{pk}(m;r)$.

- D is the decryption algorithm. It takes as input the secret key sk and a ciphertext c. Using this it produces $m = D_{pk}(c)$.

One particular group of public key cryptosystems is ElGamal-style cryptosystems.

Consider the group $\mathbf{Z}_p^*$, i.e., the multiplicative group of integers modulo p, where p is a prime. Let q be a prime, such that q divides p-1. Then there is a cyclic subgroup $\mathbf{G}_q$ of $\mathbf{Z}_p^*$ with order q. Let g be a generator for this group, i.e., $<g> = \mathbf{G}_q$.
The key generation algorithm picks primes q, p and a generator g as described above. It selects at random an element $x \in \mathbf{Z}_q$ and computes $h = g^x \bmod p$. It outputs public key pk = (q,p,g,h) and secret key sk = x.
To encrypt a message $m \in \mathbf{G}_q$ the encryption algorithm picks a random $r \in \mathbf{Z}_q$ and returns ciphertext $c = (u,v) = E_{pk}(m;r) = (g^r \bmod p, h^r m \bmod p)$.
The decryption algorithm on a ciphertext c = (u,v) returns $m = D_{sk}(c) = vu^{-x} \bmod p$.

Another variant of the ElGamal cryptosystem uses the group $\mathbf{Z}_{n^2}^*$, where n = pq, and p,q are large primes. The multiplicative group $\mathbf{Z}_{n^2}^*$ of elements computed modulo n^2 has order n*lcm(p-1,q-1), and the element (1+n) has order n in $\mathbf{Z}_{n^2}^*$.
Here the key generation algorithm outputs two elements g,h of order lcm(p-1,q-1), i.e., pk = (n,g,h) and the secret key is sk = x, such that $h = g^x \bmod n^2$.
To encrypt a message $m \in \mathbf{Z}_n$, the encryption algorithm picks a random r and computes ciphertext $c = E_{pk}(m;r) = (g^r \bmod n^2, h^r(1+n)^m \bmod n^2)$.
On ciphertext c = (u,v) the decryption algorithm outputs $m = D_{sk}(c) = ((vu^{-x} \bmod n^2) - 1)/n$.

Please note that ElGamal cryptosystems are homomorphic. I.e., $E_{pk}(m1+m2;r1+r2) = E_{pk}(m1;r1) * E_{pk}(m2;r2)$.

Common for ElGamal-style cryptosystems is that we can secret share the secret key. This means that we can have several parties that each get a share of the secret key, and only by cooperating can they perform the decryption operation. This is important in voting, where we want to have strong security guarantees that no single party is capable of decrypting a ciphertext containing a voter's vote.

There are several methods for doing this secret sharing; here we focus only on a simple linear method. Let the secret key be x. We pick at random s1,...,sk such that x = s1 +...+ sk. Give each party S1,...,SK the secret share s1,...,sk, they now have a sharing of the secret key, but no proper subset of the parties can compute the secret key.

As a step in decrypting the ciphertext c = (u,v) we want to compute u^x (we will from now on not be explicit about the group we are working in, it can be modulo p, modulo n^2, or a completely different type of group, for instance one based on elliptic curves). The parties S1,...,Sn can cooperatively do so. They simply compute $u1 = u^{s1},\ldots,uk = u^{sk}$, and publish their decryption shares. Now, anybody can compute $vu^{-x} = v(u1*\ldots*uk)^{-1}$, and from that extract the message.

There is a problem though. Imagine a party Si cheats and supplies an incorrect decryption share. In that case, we may end up with believing that the plaintext is something completely different from the message that was actually encrypted.
To solve this we let the key generation algorithm compute verification keys $h_1 = g^{s1},\ldots,h_k = g^{sk}$, and output these together with the public key. We now demand that each server S_i makes a zero-knowledge proof that u_i has been computed with the same exponent s_i as has been used to compute h_i. We will explain the notion of zero-knowledge proofs later, for now let us say that it proves the correct use of exponent s_i, without revealing anything about s_i.

Commitments

A commitment scheme consists of three algorithms K, C, and V.

- K is a key generation algorithm that outputs a public key pk.
- C is a commitment algorithm. It takes as input the public key pk and a message m. It outputs a commitment $c = C_{pk}(m)$. C is a randomized algorithm, and when needed we write the random bits used as r, and have $c = C_{pk}(m;r)$.
- V is a verification algorithm that outputs accept or reject. It takes as input a public key pk, a commitment c, an opening (m,r). It outputs accept if and only if $c = C_{pk}(m;r)$.

For the algorithms K, C, V to constitute a commitment scheme, we require that the commitment is hiding and binding.
Hiding means that from a commitment c it must be infeasible to tell which message m is inside it. Hiding comes in two flavors, computational hiding and the stronger statistical hiding. A commitment is statistically hiding, when even given infinite computing power it is still impossible to tell anything about the message inside the commitment.
Binding means that it is impossible to find a commitment c and two different openings (m1,r1) and (m2,r2) such that the verification algorithm will accept both openings. Also the binding property comes in two flavors, computational and statistical. A commitment is statistically binding if even with infinite computing power it is impossible to form a commitment c that can be opened in two different ways.

It is a fact from the cryptographic literature that a commitment cannot both be statistically hiding and statistically binding at the same time. It is possible to have commitments that are statistically binding and computationally hiding, and in fact, the ElGamal cryptosystems mentioned above are examples of such commitments. In the following, we present three examples of statistically hiding and computationally binding commitments.

Consider again the group $\mathbf{Z}_p^*$, and the cyclic subgroup G_q of order q. Let g,h be two randomly chosen generators for this group, i.e., $<g> = <h> = G_q$. The public key output by the key generation algorithm is pk = (q,p,g,h).
To commit to a message $m \in \mathbf{Z}_q$ we pick at random $r \in \mathbf{Z}_q$, and let the commitment be $c = g^r h^m \bmod p$.
An opening of the commitment c consists of (m,r), and V outputs accept if and only if $m \in \mathbf{Z}_q$, $r \in \mathbf{Z}_q$, and $c = g^r h^m \bmod p$.

Another example of a commitment scheme is the following integer commitment scheme. We use the group $\mathbf{Z}_n^*$, where $n = pq$ is a product of two primes, such that p-1 and q-1 do not have any small odd divisors. The key generation algorithm picks two squares g,h in $\mathbf{Z}_n^*$ at random.
To commit to an integer m, select r as a random 2|n|-bit number and compute the commitment $c = C_{pk}(m;r) = g^r h^m \bmod n$.
An opening of the commitment consists of (b,m,r) such that b is a square root of 1, and $c = bg^r h^m \bmod n$.

A third example of a commitment scheme is the following. We have some cyclic group **G** and select four random generators g_1, g_2, h_1, h_2 for it. The public key is $pk = (g_1, g_2, h_1, h_2)$.
To commit to a message $m \in G$, pick r_1, r_2 at random and let the commitment be $c = (u,v) = C_{pk}(m;r_1,r_2) = (g1^{r1}g2^{r2}, h1^{r1}h2^{r2}m)$.
The opening is (m,r_1,r_2), the verification algorithm checks that $c = (g1^{r1}g2^{r2}, h1^{r1}h2^{r2}m)$.

An important property of all the above examples of commitment schemes is that they are homomorphic. I.e., $C_{pk}(m_1+m_2;r_1+r_2) = C_{pk}(m_1;r_1) * C_{pk}(m_2;r_2)$, or if we prefer multiplicative notation for the latter commitment's message space, we have $C_{pk}(m_1*m_2;r_1+r_2,s_1+s_2) = C_{pk}(m_1;r_1,s_1) * C_{pk}(m_2;r_2,s_2)$.

We can easily extend the commitments to commit to several values at once. Let the public key consist of $g,h_1,\ldots,h_n$. Then we can commit to $m_1,\ldots,m_n$ as $c = g^r h1^{m1}\ldots hn^{mn}$.

CLAIM
The following variation of an integer commitment scheme.

Let $n = pq$ be the product of two primes p and q. Let furthermore, p',q' be two primes dividing respectively p-1 and q-1. Reasonable sizes are |p|=|q|=1500 bits and |p'|=|q'|=120 bits. Both p,q,p' and q' must be kept secret. Let furthermore, t be an integer such that $t > |p'|+|q'|$. For instance we could with the above parameters selects $t = 300$.
Pick at random g,h such that <g>=<h> are groups of order p'q'.
The key generation algorithm outputs the public key pk = (n,g,h,t).

To commit to an integer m, pick at random r as a t-bit number. Compute the commitment $c = C_{pk}(m;r) = g^r h^m \bmod n$.

To open the commitment reveal the opening (m,r). The verification algorithm on opening (m,r) checks that $c = g^r h^m \bmod n$.

Variations of the scheme:
As mentioned before it is possible to make a variation of the integer commitment scheme that allows for commitment to multiple integers at once.
One can select p',q' such that they are composites. It is important, however, that they are selected such that it is hard to guess a number N such that p'|N or q'|N.

Note that we deliberately work in a moderately small subgroup of Z_n^* in order to gain better efficiency.
This has potential use in both voting protocols and many other cryptographic protocols.

Zero-knowledge proofs

A zero-knowledge proof or zero-knowledge argument is an interactive protocol to be run between two parties (or in some cases more parties). We call them respectively the prover and the verifier. Both of them know some common input x, and now the prover wants to convince the verifier that x has some particular property, for instance that there exists a witness w such that (x,w) belongs to some NP-language. To do so, they exchange messages according to the zero-knowledge protocol, and in the end, the verifier decides whether to accept or reject the statement.

We call such an interactive protocol a zero-knowledge argument if it has the following three properties

- Completeness: If the prover knows a witness w for the property of x, then he can make an honest verifier accept.
- Soundness: If the statement is false, i.e., no such w exists; any (possibly cheating) prover cannot make an honest verifier accept.
- Zero-knowledge: Any (possibly cheating) verifier does not learn anything but the veracity of the statement from interacting with an honest prover.

There are many variations of how to define zero-knowledge proofs and arguments. Among them are non-interactive variants, where we instead assume a common reference string, chosen with some particular distribution, is available to both prover and verifier. Non-interactive zero-knowledge proofs and arguments are publicly verifiable.

Another variation is honest verifier zero-knowledge, where the zero-knowledge property holds if the verifier follows the protocol, but may not hold if the verifier deviates from the protocol. A stronger version of this is special honest verifier zero-knowledge, where the verifiers messages are public coin (i.e., consists of uniformly random bits) and where it is possible to simulate the entire proof (without knowledge of the witness w) if we are given in advance the messages (challenges) that the verifier sends.

A popular method for making a special honest verifier zero-knowledge proof non-interactive is the Fiat-Shamir heuristic. In the Fiat-Shamir heuristic, we compute the challenges as suitable hash-values. This means that we do not need a verifier to choose the challenges.

Mix-nets

Suppose we have a bunch of ciphertexts c1 = (u1,u2) = E_{pk}(m1),...,cn = (un,vn) = E_{pk}(mn). We want to learn the messages m1,...,mn, but in a random order, we do not want anybody to be able to link messages and ciphertexts.

A group of servers cooperating to do so is called a mix-net. Using ElGamal-encryption, we can construct a mix-net in a simple manner. Using the secret sharing described before the servers S1,...,Sk each have a share s1,...,sk of the secret key such that x = s1+...+sk.

Server S1 peels off the layer of encryption corresponding to its own secret share, it rerandomizes the ciphertexts and outputs them in a permuted order. I.e., S1 selects a permutation π, randomizers R1,...,Rn and outputs $(U1=g^{R1}u_{\pi(1)}, V1=(h2*\ldots*hk)^{R1}v_{\pi(1)}\,u_{\pi(1)}^{-s1}),\ldots,(Un=g^{Rn}u_{\pi(n)}, Vn=(h2*\ldots*hk)^{Rn}v_{\pi(n)}\,u_{\pi(n)}^{-s1})$.

Server S2 peels off another layer of the encryption corresponding to its secret share. I.e., if we call the output from S1 (u1,v1),...,(un,vn), then it selects a permutation π, randomness R1,...,Rn and outputs $(U1=g^{R1}u_{\pi(1)}, V1=(h2*\ldots*hk)^{R1}v_{\pi(1)}\,u_{\pi(1)}^{-s1}),\ldots,(Un=g^{Rn}u_{\pi(n)}, Vn=(h3*\ldots*hk)^{Rn}v_{\pi(n)}\,u_{\pi(n)}^{-sn})$.

When the last server Sk peels off a layer of the encryption, then V1,...,Vn constitute a permutation of m1,...,mn. More precisely, if we call the permutations selected by S1,...,Sk for π1,...,πk, and let π(•) = π1(...(πk(•)...), then we have $V1=m_{\pi(1)},\ldots,Vn=m_{\pi(n)}$. However, only if all servers cooperate will they know π and be able to link messages to their ciphertexts. Conversely, if just a single server is honest, then the permutation is secret.

Mix-nets are useful in voting, since it allows encrypted votes to be decrypted and permuted. This way votes can be counted, but at the same time, nobody can link voters with their votes.

Shuffle-and-decrypt

The problem in the above mix-net is how to avoid that one of the servers replaces encrypted votes with ciphertexts containing false votes. One possible solution to this problem is to let each server make a zero-knowledge argument of correctness of the shuffle-and-decrypt operation it performs.

I.e., call the input (u1,v1),...,(un,vn) and the output (U1,V1),...,(Un,Vn). Furthermore, let g,h and H be public.
The prover has private input π,R1,...,Rn and s, such that $h = g^s$ and $(U1=g^{R1}u_{\pi(1)}, V1=H^{R1}v_{\pi(1)}u_{\pi(1)}^{-s}),\ldots,(Un=g^{Rn}u_{\pi(n)}, Vn=H^{Rn}v_{\pi(n)}u_{\pi(n)}^{-s})$.

CLAIM
The following method to demonstrate that indeed (u1,v1),...,(un,vn), (U1,V1),...,(Un,Vn), g,h,H is on the form described above, without revealing π, R1,...,Rn and s.

We need additional public data in form of public keys for three types of commitment schemes. We omit explicitly writing the public keys, and simply write respectively mcom, commit, and COM for the three commitments.

Multi-commitment mcom is used for committing to multiple messages at once, in our case n messages. Furthermore, it has a homomorphic property. I.e., mcom(m1+M1,...,mn+Mn;r+R) = mcom(m1,...,mn;r) * mcom(M1,...,Mn;R).

Commitment scheme commit is used for committing to a single element at a time. It too has a homomorphic property commit(m+M;r+R) = commit(m;r) * commit(M;R).

Finally, we use a base commitment scheme COM, where the homomorphic property is COM(mM;r+R,s+S) = COM(m;r,s) * COM(M;R,S).

The protocol proceeds in 7 steps:

1. The prover picks rs at random and computes cs = mcom($\pi(1),\ldots,\pi(n)$;rs). He sends cs to the verifier.
2. The verifier picks at random t1,...,tn and sends them to the prover.
3. The prover picks rt at random and computes ct = mcom($t_{\pi(1)},\ldots,t_{\pi(n)}$;rt). He sends ct to the verifier.
4. The verifier picks at random λ,x and sends them to the prover.
5. The prover computes the following:
   For j=1 to n: aj = $(\pi(1) + \lambda t_{\pi(1)} - x) * \ldots * (\pi(j) + \lambda t_{\pi(j)} - x)$.
   Picks d1,...,dn and rd at random and sets cd = mcom(d1,...,dn;rd).
   Picks d at random and sets $D = g^d$.
   Picks r1,...,rn at random and computes c_1 = commit(d_1;r_1), c_2 = commit(d_2a_1;r_2), ..., c_n = commit(d_na_{n-1};rn). Picks r at random and sets c = commit(0;r).
   Picks R at random and computes $U = g^R U_1^{d1} * \ldots * U_n^{dn}$, $V = (hH)^R V_1^{d1} * \ldots * V_n^{dn} (U_1^{d1} * \ldots * U_n^{dn})$, and $W = (U_1^{d1} * \ldots * U_n^{dn})^d$.
   Picks Rv and Rw at random and sets Cv = COM(V;Rv) and Cw = COM(W;Rw).
   He sends cd, D, c1, ..., cn, c, U, Cv, Cw to the verifier.
6. The verifier picks at random ε, e and sends them to the prover.
7. The prover computes the following:
   $f_1 = e(\pi(1) + \lambda t_{\pi(1)} - x) + d1, \ldots, f_n = e(\pi(n) + \lambda t_{\pi(n)} - x) + d_n$, and $zf = rs + \lambda rt + rd$.
   $z = r - ef_2 \ldots f_n r_1 - \ldots - e^n r_n$.
   $Z = R - e(R_1(\pi(1) + \lambda t_{\pi(1)} - x) + \ldots + R_n(\pi(n) + \lambda t_{\pi(n)} - x))$.
   $f = \varepsilon s + d$.
   $Zv = \varepsilon Rv + Rw$.
   He sends $f_1,\ldots,f_n$,zf,z,Z,f,Zv to the verifier.

The verifier accepts if all elements belong to the correct groups and have the correct size, and the following checks pass:

Set cx = mcom(x,...,x;0) and check that mcom(f1,...,fn;zf) = $(c_s c_t^{\lambda} c_x^{-1})^e c_d$.

Check that commit($e^{n+1}a_n - ef_1 \ldots f_n$;z) = $c_1^{-ef2\ldots fn} \ldots c_n^{-e\ldots e} c$.

Check that $g^Z U_1^{f1} * \ldots * U_n^{fn} = (u_1^{1+\lambda t1-x} \ldots u_n^{n+\lambda tn-x})^e U$.

Check that $g^f = h^{\varepsilon} D$.

Check that COM($(V_1^{f1} \ldots V_n^{fn})^{\varepsilon} (U_1^{f1} \ldots U_n^{fn})^f (hH)^{\varepsilon Z} (v_1^{1+\lambda t1-x} \ldots v_n^{n+\lambda tn-x})^{-e\varepsilon}$;Zv) = $C_v^{\varepsilon} C_w$.

The protocol above is public coin, complete, sound and statistical honest verifier zero-knowledge.

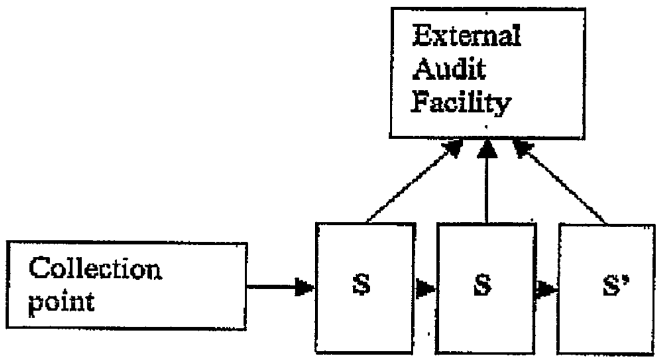
Using techniques from [GMY] it is a simple matter to make it statistical zero-knowledge. Using the Fiat-Shamir heuristic, i.e., computing the challenges $t_1,\dots,t_n$, λ, x, ε and e as suitable cryptographic hashes it is easy to make the protocol non-interactive. This way it can also be made publicly verifiable.

Using randomization it is possible to speed up the verification process, see [DGS] and [BGR] for comments on batching techniques. It is furthermore well known that various techniques for fast multiple exponentiation exist.
For instance we can pick γ at random and check whether COM($(V_1^{f1}\dots V_n^{fn})^{e}(U_1^{f1}\dots U_n^{fn})^{f+\gamma}(hH)^{eZ}(v_1^{1+\lambda t1-x}\dots v_n^{n+\lambda tn-x})^{-\varepsilon e}g^{\gamma Z}(u_1^{1+\lambda t1-x},\dots u_n^{n+\lambda tn-x})^{-\gamma e}U^{-\gamma};Zv)$ $= C_v^{\varepsilon}C_w$. This saves n exponentiations.

Efficient proofs for proving correctness of decryption are well known in the cryptographic literature. Likewise, many proofs of correctness of a shuffle exist [FS,G,NEF01,Npatent], embodiments of our proposed shuffle-and-decrypt proof are zero-knowledge and more efficient than previous proofs.
Shuffle-and-decrypt proofs can be used in anonymization protocols, voting protocols is one particular instance of protocols where anonymization is needed.

Optimized MIX nets

A traditional MIX net consists of a number of shuffle servers, each refreshing the randomness part of the encryption of encrypted votes, each permuting the votes and each producing a zero-knowledge proof that their output is a permutation and re-encryption of the input. In the final step the votes must be decrypted and zero-knowledge proofs must be included that the votes have been decrypted correctly. The correctness of the result of the election can be verified by an external audit facility, which verifies correctness of the counting by inspecting the input, output and zero-knowledge proofs of each server.



A Mix net. The S servers re-encrypt (refreshes randomness) and permutes votes, whereas the S' server decrypts votes. All provide zero-knowledge proofs that they have done their tasks correctly.

It is not desirable that the private key used for decryption is in the possession of only one entity. Therefore S' should consist of several entities, which secret-share the private key of the election. However this solution is impractical.

The [DGS] crypto system is special since it is ElGamal style. It is thus possible to share the secret key **X** as a sum of keys $\mathbf{X}_1 + \ldots + \mathbf{X}_m$, where each $\mathbf{X}_i$ is known to one shuffle ($\mathbf{h} = \mathbf{g}^X$ in the system, the pair **(g,h)** is the public key, whereas **X** is the secret key).

Each shuffle can then partially decrypt, re-randomize and make a shuffle proof.

The mechanism for partial decryption and re-randomization is:

$(g,h,g^r,(1+n)^v h^r) \to (g,hg^{-X}{}_1, g^{r+R}, (1+n)^v (h\, g^{-X}{}_1)^{r+R})$,

where v is the vote including marking with correspondence to a manual vote, r is the original randomness, R is new randomness, g is a generator, $h = g^X$ and (1+n) is a special element generating a group with an efficient discrete logarithm.

We will arrange embodiments of our voting system such that each shuffle partially decrypts the votes using its share of the secret key. The final server completes the decryption of the votes and produce zero-knowledge proofs of the correctness of the decryptions. In this way we will not need additional entities in order to perform the decryption securely. Two different types of embodiments using this type of encryption are possible.

- Embodyments where the shuffles perform zero-knowledge proofs of the correctness of their actions and the keys for the input and output encryptions are different.
- Embodyments where the verification of correctness of votes and the computation of the result is done out of band using homomorphic encryption properties. We will provide an example embodiment of our invention of this type. (In this case the maximal security is obtained with 3 shuffle servers and a server for decrypting the result. Three servers need to cooperate to break the secrecy in this case. We do not consider this to be a large problem since three shuffle servers is the natural choice).

A naïve MIX net implementation is not very fast. However, doing zero-knowledge proofs out of band of the shuffles and using other, generic, optimizations, it is possible to increase the throughput dramatically. We list a number of optimizations:

- Partially decrypting in each shuffle and not providing zero-knowledge proofs gives a factor of about 3.
- Partial decryption and rerandomization of votes can be parallelized arbitrarily. This gives an improvement of performance by a factor of 5-10.

- The order in which the servers process each vote need not be the same for all votes. For example if there are three servers performing re-encryption, the votes can be distributed in three pools depending on their election district (since the result will normally be specified out for election districts, permutations between election districts are not relevant) and the pools are rotated between the re-encrypting servers until each vote has been once at each server. This gives a factor of about 3 compared to naively letting the first server finish its work before the next server starts.
- If g is chosen in a subgroup of small order with elements that are indistinguishable from elements of the whole of Z_{n*n}*, randomness and keys may be chosen shorter. Such optimizations are known for ElGamal over a prime and are also possible with ElGamal over a RSA modulus. It may give a factor 2-4 depending on the size of the RSA modulus of the crypto system.

All in all this means that detachment of identities from votes can be performed about 45-90 times faster than for a naïve MIX net implementation. The final decryption of votes can also be parallelized arbitrarily.

Attacks against the Scheme.

In order for a MIX net to have optimal security properties it is necessary that each shuffle server verifies the zero-knowledge proofs of the predecessors before it performs its own MIX. As we have discussed this is not optimal with respect to performance, so it is fair to provide an account of the attacks made possible by not letting this verification take place.

If we count the votes by an out of band method, we can be sure that it will be discovered if the result of the election is altered. In the embodiment we will provide such a count is done securely using a homomorphic count. Thus we will have full security when it comes to making sure that the result of the election is correct.

However, some attacks against the secrecy of the election are possible. Since the crypto-system is homomorphic, the first S server can add numbers to votes and it can multiply the votes by a constant factor. This can normally be done in a way such that the vote as well as the number added can be separated from each other when the vote is decrypted. We will say that the encrypted votes are marked. Depending on which servers the first S server cooperates with, different properties of the attack are possible.

- If the first S server acts alone, the decrypting server will discover the fraud but also be presented for the association between identities of voters and votes cast. If the decrypting server is honest, not compromised and checks whether votes are correctly formatted before they are published, the anonymity of the election will not be broken. Further, the fraud will be detected and a delayed count can take place with the first S server replaced.
- If the first S server and the decrypting server work together, they will together be able to break the secrecy of the election completely. Because of the zero-knowledge proofs of correct decryption of votes, the fraud will be detected.

- If the first S server and the decrypting server work together with all external audit facilities used, the abovementioned fraud need not be detected. (The decrypting server can in this case clean the votes before it publishes them and provide wrong zero-knowledge proofs that the audit facilities will let through undetected.)
- If the first S server and the decrypting server work together with the last S server, they will be able to break the secrecy together without being detected. (The decrypting server decrypts votes, sends them back to the last S server, which cleans its encrypted output for the marks and submits a new, correct output.)

The basic properties are that two servers need to cooperate in order to break the secrecy, while accepting that their fraud will be detected. Three entities need to work together in order to break the secrecy without being detected. This can be improved on slightly by letting either the first or the last S server carry out a shuffle proof.

In the case, where we have two S servers and one decrypting server we see that there is no real loss of security. The two S servers could anyway break the secrecy by interchanging permutations. The first attack also has the equivalent that one of the S servers submits its permutation in clear text to the other S server, this will be discovered by the other S server, which will (unwillingly) be able to break the secrecy.

Write-In Candidates

In the US and some other countries it is common to use write-in candidates. That means that it is possible to vote for a candidate not on the list. This cannot be ignored for embodiments of systems to be applied in practice.

MIX nets can handle write-in candidates without problems, whereas homomorphic encryption can't deal with write-in candidates. Below we describe how homomorphic 'encryption' can however be used to prove that a list of write-in candidates is correct. First we give some background on commitment systems:
A verification system is a computationally hiding commitment system that is further supplied with a private key that breaks the computationally hiding property without breaking the commitment system properties. That means that a person in possession of a secret key X for the verification system will be able to verify a claim that a given commitment contains a given message without being provided with an opening of the commitment. However, knowledge of X will not provide any knowledge at all about the cipher-text space of the commitment system. In particular, the cipher-text space observed by a person with knowledge of X may appear to be an infinitely large space like **Z** just as if the person had not been in possession of X.

A homomorphic verification system is a verification system for which the underlying commitment system is homomorphic.

Example: Consider $\mathbf{Z}_n$, where $\mathbf{n}$ is an RSA modulus with unknown factorization. Pick generators $\mathbf{f}$ and $\mathbf{h}$ of $\mathbf{Z}_n^*$. and set $\mathbf{g} = \mathbf{f}^X$ for a randomly chosen $\mathbf{X}$. We define the ElGamal style homomorphic system

$$\mathbf{V(m;r) = (f^r, h^m g^r)}$$

Then $\mathbf{V}$ is a homomorphic commitment system and $\mathbf{X}$ is the secret key that breaks the computationally hiding property.

Please notice that $\mathbf{V}$ is not a crypto system. The discrete logarithm in $\mathbf{Z}_n$ cannot be computed efficiently, so decryption is impossible if $\mathbf{n}$ is a large RSA modulus. In fact, if decryption were possible in general, the real message space would be known, which would imply breaking the RSA modulus, which is clearly not possible from the information given. Also in this way we see that the message space is $\mathbf{Z}$, so the basic properties of the commitment systems are preserved.

In short we observe that this system has the property that the message space is all of $\mathbf{Z}$, that the system is computationally hiding for an adversary without knowledge of the secret key, but entities with knowledge of the secret key are able to verify a claim efficiently that a commitment is a commitment to a particular value. Further, the private key can be secret shared like for other ElGamal style systems.

We remark that cryptographic primitives with the same properties as verification systems but without the homomorphic property are easy to construct from standard cryptographic primitives. For example one can take a hash function H with 16 byte output, consider the hash value **H(m)** as an AES key, use this key to encrypt a fixed value and finally encrypting the result using a public RSA key. This primitive allows persons in possession of the corresponding private RSA key to verify whether it was computed on a fixed value whereas it is computationally hiding for persons not in possession of the private key. Such a primitive could for example be used for time-stamping systems that allow only particular entities to verify time-stamps.

One novel aspect of embodiments of our invention of homomorphic verification systems is therefore the ability to verify several claims in one combined operation while keeping some properties of the individual claims secret. In the novel applications for voting systems we shall see, it will be the origin of the individual messages.

Claim: Use of homomorphic verification systems for verifying that messages $\mathbf{m}_1,\ldots,\mathbf{m}_k$ are authentic without revealing their origin in the following way: The entities $\{\mathbf{E}_j\}$ producing the messages each choose a large random numbers $\mathbf{e}_j$, $\mathbf{r}_j$ and ρ_j. They submit $\mathbf{m}_j$, $\mathbf{V}(\mathbf{e}_j, \mathbf{r}_j)$ anonymously to one entity (entity **A**) and $\mathbf{V}(\mathbf{m}_j\, \mathbf{e}_j, \rho_j)$ to another entity (entity **B**) in such a way that it is properly authenticated. The authenticity of the $\{\mathbf{m}_j\}$ is verified by having entity **B** submitting $\Pi\, \mathbf{V}(\mathbf{e}_j, \mathbf{r}_j)^{\mathbf{m}_j}$ to entity **A**, which computes $\mathbf{C} = \Pi\, \mathbf{V}(\mathbf{m}_j\, \mathbf{e}_j, \rho_j)^{-1}\mathbf{V}(\mathbf{e}_j, \mathbf{r}_j)^{\mathbf{m}_j}$. Finally a trusted entity, which knows **X**, verifies that **C** is a commitment to zero.

Let **E** denote a homomorphic crypto system. The implementation in the context of a voting system can be done as follows (for simplicity we use **V** as commitment system also, in practice some commitments would be done in a simpler system, which is preferably statistically hiding):

- Let **v** be the vote, $v=\Sigma\delta j\ M^j$. Some indices **j** represent write-in votes, whereas others represent candidate or list votes. $M = p^2$ is a square of a prime larger than the number of voters. (See [DGS], [DJ01]).
- Submit **E(v)**, $V(\Sigma\delta j\ p^j)$ together with a non-interactive zero-knowledge proof of equivalence between the two and a non-interactive zero-knowledge proof that the vote conforms with the rules of the election (see [DGS], [DJ01]).
- Let **m** be a write-in vote corresponding to index **k**. Submit **E(m)**, **V(m)** together with a non-interactive zero-knowledge proof of equivalence of **E(m)** and **V(m)** and a non-interactive zero-knowledge proof that either $m=0$ or $\delta_k = 1$. (This can be done by decomposing $V(\Sigma\delta j\ p^j)$ into two commitments $v_1 = V(\delta_k\ p^k)$ and $v_2 = V(\Sigma j \neq k\ \ \delta j\ p^j)$, proving that either v_1 or $V(-p^k)\ v_1$ is a commitment to zero and proving that the content of either **V(m)** or $V(-p^k)\ v_1$ is a commitment to zero. Such proofs are standard.)
- Pick a random number e_m and submit $V(e_m)$ and $V(me_m)$ together with a multiplication proof that the content of $V(me_m)$ is the product of the content of $V(e_m)$ and **V(m)**.
- Sign the entire vote including all proofs.

Notice that the number e_m will never become known to anybody since no encryption of it is submitted.

Now say that we count the votes by using the homomorphic property and decrypt the result. By using the homomorphic property we get the encryption:

$$V_1 = V(\Sigma\ me_m) = \Pi\ V(me_m).$$

If we also use a 'MIX net', we get the individual numbers **m** and $V(e_m)$ coupled in pairs but detached from the identities of the voters. Thus we may compute

$$V_2 = V(\Sigma\ me_m) = \Pi\ V(e_m)^m.$$

Using the secret key **X**, secret shared between the same persons that share the private key for the crypto system (homomorphic sharing, not the sharing between 'MIX' net servers), we can check that V_1 and V_2 have the same content. If the e_m were chosen large and random, there is in practice no way to fake this.

Claim: Use a hormorphic crypto system, for example the above mechanism for using a homomorphic property to check the write-in votes coming out from a MIX net.

Claim: Also use a hormorphic crypto system, for example the above mechanism in the way that all votes are treated as written-in votes. I.e. there are no zero-knowledge proofs of correctness of normal votes, only use of this mechanism.

Claim: Also use a hormorphic crypto system, for example the above mechanism for verifying correctness of encrypted information linking electronic ballots to paper ballots. (The information linking electronic votes to paper voters take the role of **m**. Similar to above except that in this case there is no proof linking **m** to an ordinary vote.)

Attacks against the Scheme.

If the machines from where voters vote leak the $\mathbf{e}_i$ it may be possible for the last shuffle server and the decrypting server together to produce different **m**'s. Notice however that this requires three cooperating entities and will with a high probability be detected by the tests against paper ballots in our invention.

Also some attacks against the secrecy of the election are possible. The first S server can replace $\mathbf{E}(\mathbf{m}_i)$ and $\mathbf{V}(\mathbf{e}_i)$ in some ways:

- Replaced by $\mathbf{E}(\mathbf{m}_i)^{1/2}$ and $\mathbf{V}(\mathbf{e}_i)^2$ or by higher powers of ½ and 2. If write-in votes are published no matter whether they make sense or not, or if the first **S** server works together with the decryption server, this can be used to check what individual voters voted. This will be detected unless further the last **S** server is also involved in the fraud.
- Replaced by $\mathbf{E}(\mathbf{m}_i)^2\mathbf{E}(-\mathbf{m}_0)$ and $\mathbf{V}(\mathbf{e}_i)$. This can be done and will pass all tests provided that the first **S** server correctly guesses the content of each vote it tampers with.

The last attack is potentially rather serious because the first **S** server can be buying votes and use it for verifying that the vote-sellers deliver. Consequently, as long as vote-sellers are honest this vote-buying will not be detected. However, if a vote-seller does not deliver, the fraud will be detected. This attack (and similar ones with different powers than 2) can be made infeasible by including a few random bits in each vote at a specific location.

Again we conclude that whereas this is not quite as secure as a MIX net where all shuffle-proofs of predecessors are verified before the next shuffle server starts, attacks against the integrity of the election require three cooperating entities and will be detected with a high probability whereas attacks against the secrecy of the election with potential for not being discovered require at least two cooperating entities if the votes are enhanced with a few random bits.

Signing Votes

We remark that embodiments of the inventions claimed are possible without using digital signatures, so the scope of the claims in this document is independent of digital signatures. We will nevertheless motivate the approach we have taken in our embodiments.

It is essential that encrypted votes are digitally signed such that there is 100% accountability about *exactly* where each electronic vote came from. Some pilot systems have attempted to use chip cards[3] for that purpose, but face the difficulty that chip cards are expensive and that chip cards with signing keys are not widespread.

The alternatives to using a portable device like a chip card to store the private keys of the voter on are to:

- Not store the private key.
- Store the private key in a non-portable device.

The first option is taken in the e-Vote project, where the Internet-voting pilot system works in the way that a public/private key pair is generated in an applet running on the computer of the voter. A certificate on the public key is then issued on the fly based on credentials that the voter receives by mail such that the vote can be properly signed. Depending on the procedures applied for distributing the credentials and the properties, configuration and operation of the on-line CA, this may be a legally binding signature. By using this mechanism the e-Vote system is optimal in the sense that it uses the simplest and cheapest possible mechanism for creating legally binding signatures on encrypted electronic votes.

For an e-voting system that takes place at election sites it is however unacceptable that the devices used for casting votes store the private key of the voter (when also, supposedly, only for a short time). The remaining option is thus to store the private key in another, non-portable device. Such a device - which we call the Signer - is described in our patent application PCT/GB02/03707. The Signer will then be operated at central locations different from election sites. Digital signatures are produced by the Signer on the basis of credentials provided by the voters, and each digital signature is logged by the Signer.

It is strongly preferable that two-factor authentication is used for voting[4]. One factor is used in the public sphere, such that vote-buying by buying credentials can be prevented. The other factor is used in privacy when the vote is cast, such that accountability is assured. The Signer is designed to deal with two-factor authentication in a highly secure and tamper resistant way since it is distributed in two servers that each know of one factor of the authentication.

We conclude that circumstance dictate the Signer approach to be the preferred solution, both cost-efficient and secure to use. However, if the Signer is used it is sufficient that each voter receives a voter card by mail as usual with two authentication factors printed on it in order to produce digital signatures.

Use of the Signer preferably requires the e-Voting system to be on-line. However the security[5] does not rely on confidence in the device used for casting votes and printed ballots may serve as backup in case of lacking on-line availability[6].

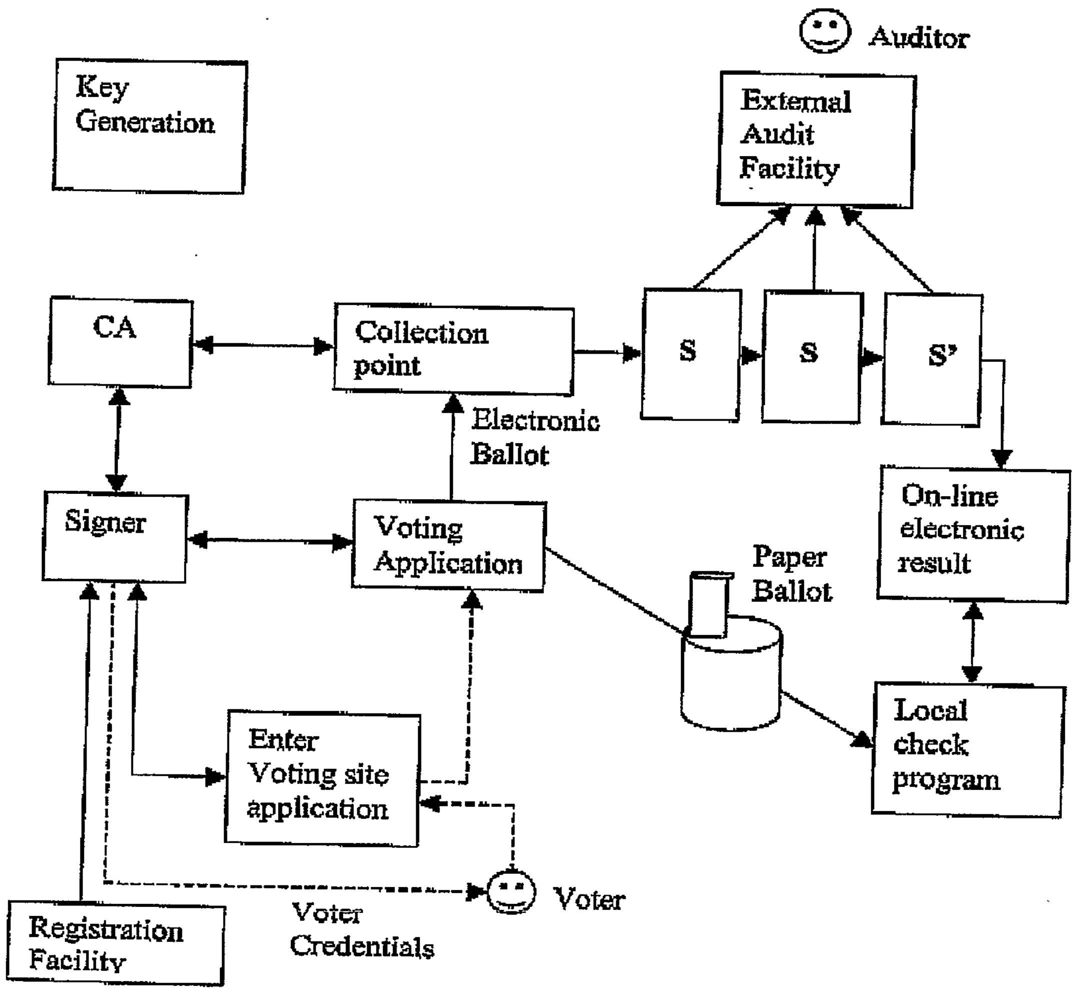
[3] The Cybervote project is an example. DRE systems also use chip cards, but in a different way that is not related to digital signatures and does not provide the accountability we are discussing.
[4] We preserve the option that the voter receives both factors in one letter. This is not really a problem since the identity of the voter can be checked when giving off the first factor (with the same level of scrutiny that is used for manual elections, that differs a lot from country to country). The central point is that vote-buying must be prevented by having a public and a private authentication factor.
[5] At least security against undesired influence on the outcome of elections.

Example Embodiments

This section describes how embodiments can be made. First we give two examples of how to encrypt information linking electronic and paper ballots

1) Enlarge homomorphically encrypted votes (generalized Paillier or Damgård-Groth such that the plain-text space is $Z_{n^{s+1}}$ instead of Z_{n^s}. Represent (vote, manual ballot) as vote + n^s (manual ballot). Project the encrypted vote to $Z_{n^{s+1}}$ before doing the zero-knowledge proof of correctness of the vote.
2) Use two homomorphic encryption keys with orders of clear-text spaces and ciphertext spaces mutually prime for letting the product have decent properties again. Do homomorphic encryption proofs in the vote space only, but do the MIX net proof in the product space (if a MIX net proof is done).

We describe a realisation below that is as simple as possible. This system is an example of a traditional MIX net system enhanced with additional information linking electronic ballots and paper ballots:



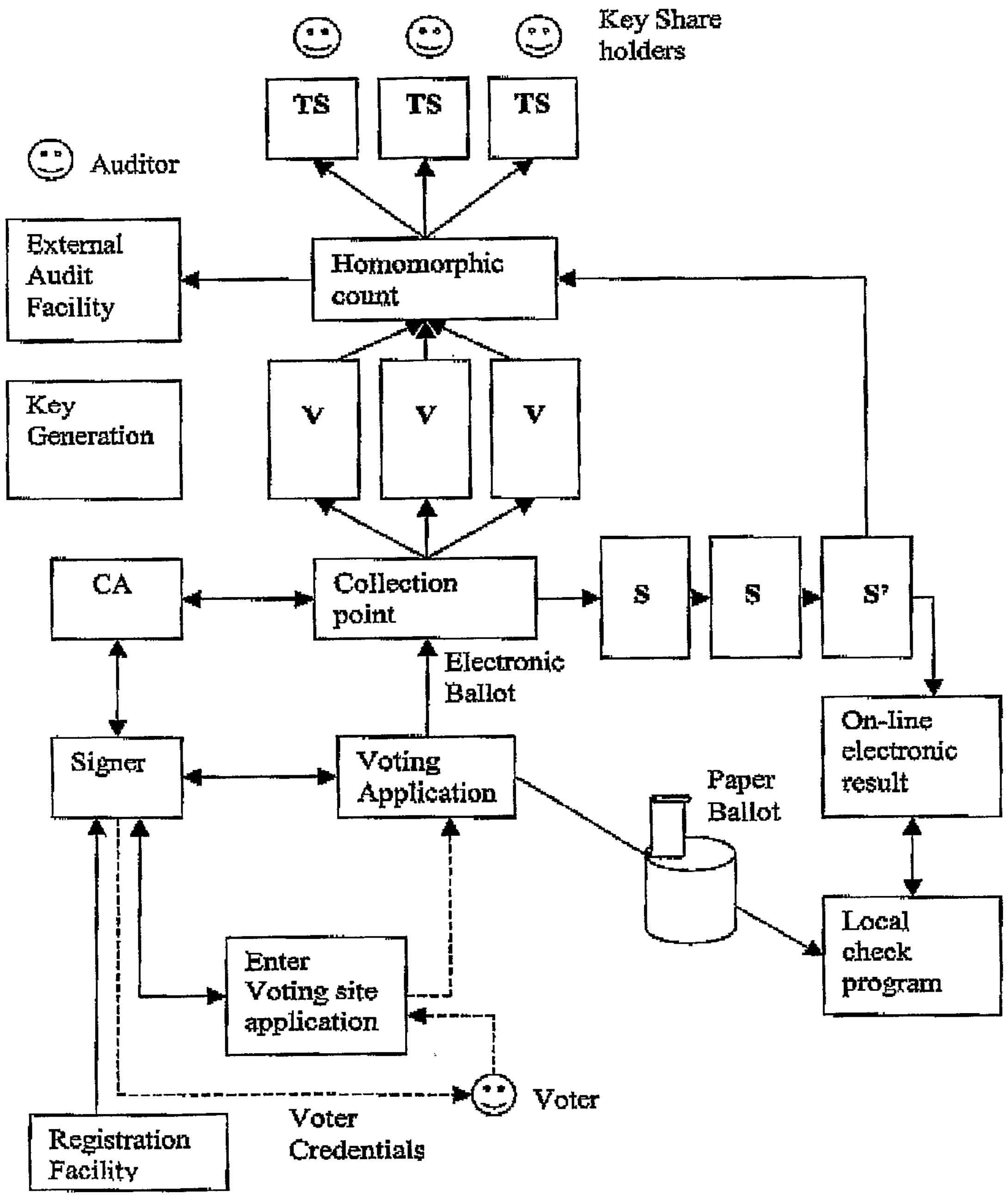
[6] I.e. in case of lacking availability of services, voters should continue to vote using printed ballots and a manual count will be necessary for the election site.

We briefly describe the individual components:

- The "Registration Facility" is a public sector system for keeping track on the eligible voters. The registration facility interfaces with the Signer for registering voters for the system.
- The Signer is the signature server referred to above, which is used for keeping track on voter credentials and voter identities in the voting system and for signing electronic votes. When voters are registered on the Signer, the Signer registers them at a CA for certification. The Signer further sends credentials to the voters and make available functions for disabling voters who cease to be eligible or loose their credentials.
- The CA issues certificates on voters.
- The "Enter Voting Site Application" accepts one credential from the voter, which is provided in public. In this way it is prevented that voters can buy credentials and bring more credentials with them into the place where votes are cast. A manual check of the voter identity is carried out when the "Enter Voting Site Application" is used. The "Enter Voting Site Application" is also responsible for handling and logging most exceptions to the normal flow of events (examples: A voter identifies himself but has lost his credentials. A voter looses his second piece of authentication inside the voting site. A voter changes his mind before submitting the paper ballot but after having submitted the electronic ballot).
- The "Voting Application" is the application/machine used for casting votes. This can for example be a touch screen machine. The voter selects his choices and gives off his second credential that is used for having the Signer signing the vote. As a result an electronic ballot and a paper ballot are created. The electronic vote is sent on-line to a collection point, whereas the voter carries the manual vote out in the public sphere, where he enters it into a traditional ballot box.
- The "Local Check Program" is a program used for checking the votes after the election is finished. (Scanning of information linking paper ballots to electronic ballots, checking correspondence with information on electronic ballots, checking that the number of paper ballots equals the number of electronic ballots and checking a selected number of ballots, presumably less than 200, with the corresponding electronic ballot.)
- The "Collection Point" is a server, which collects votes from at least one district and checks syntax and digital signatures on the votes.
- The S servers are servers holding a share of the private keys of the election. They re-encrypt and permute votes and generate a non-interactive zero-knowledge proof that they have done the job correctly.
- The S' server performs the last part of the decryption and provides a non-interactive zero-knowledge proof of correctness of the decryption of each individual vote.
- The "Key Generation Application" is an off-line application operated under particularly stringent security measures used prior to the election for generating key pairs of the election (crypto system, commitment

system). The public keys and private key parts are distributed to the relevant entities. Notice that the S servers should be operated by different organizations/persons in order to ensure secrecy of votes.

We describe a realisation below that is optimized for performance and security in the sense that performance-demanding generation of zero-knowledge proofs is done at the election sites and verification is scalable, such that all zero-knowledge proofs can be verified before the result is published:



We briefly describe the individual components:

- The registration facility is a public sector system for keeping track on the eligible voters. The registration facility interfaces with the Signer for registering voters for the system.
- The Signer is the signature server referred to above, which is used for keeping track on voter credentials and voter identities in the voting system and for signing electronic votes. When voters are registered on

the Signer, the Signer registers them at a CA for certification. The Signer further sends credentials to the voters and make available functions for disabling voters who cease to be eligible or loose their credentials.

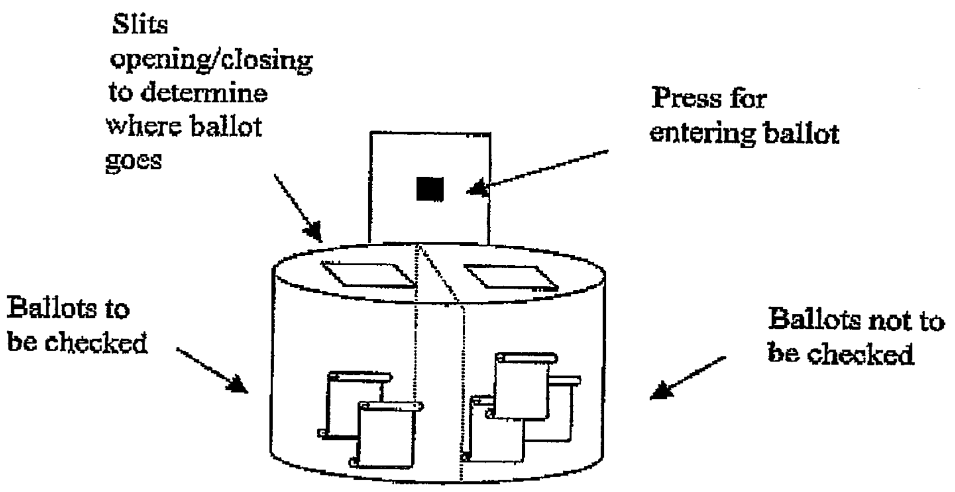
- The CA issues certificates on voters.
- The "Enter Voting Site Application" accepts one credential from the voter, which is provided in public. In this way it is prevented that voters can buy credentials and bring more credentials with them into the place where votes are cast. A manual check of the voter identity is carried out when the "Enter Voting Site Application" is used. The "Enter Voting Site Application" is also responsible for handling and logging most exceptions to the normal flow of events (examples: A voter identifies himself but has lost his credentials. A voter looses his second piece of authentication inside the voting site. A voter changes his mind before submitting the paper ballot but after having submitted the electronic ballot).
- The "Voting Application" is the application/machine used for casting votes. This can for example be a touch screen machine. The voter select his choices and gives off his second credential. As a result an electronic and a paper ballot are created. A non-interactive zero-knowledge proof of correctness of the electronic vote is attached to the electronic vote. The electronic vote is signed by the Signer using the second credential of the voter. The electronic vote is sent on-line to a collection point, whereas the voter carries the manual vote out in the public sphere, where he enters it into a traditional ballot box.
- The "Local Check Program" is a program used for checking the votes after the election is finished. (Scanning of information linking paper ballots to electronic ballots checking correspondence with information on electronic ballots, checking that the number of paper ballots equals the number of electronic ballots and checking a selected number of ballots, presumably less than 200, with the corresponding electronic ballot.)
- The "Collection Point" is a server, which collects votes from at least one district and checks syntax and digital signatures on the votes.
- The S servers are servers holding a share of the private keys of the election. They re-encrypt and permute votes (zero-knowledge proofs and the signature are removed).
- The S' server performs the last part of the decryption and provides a proof of correctness of the decryption of each individual vote.
- The "Key Generation Application" is an off-line application operated under particularly stringent security measures used prior to the election for generating key pairs of the election (homomorphic crypto system, homomorphic commitment system and homomorphic verification system). The public keys and private key parts (secret shared in two different ways) are distributed to the relevant entities. Notice that the S servers should be operated by different organizations/persons in order to ensure secrecy of votes.
- The V servers are used for verifying zero-knowledge proofs on the individual votes. Notice that one of the schemes for write in candidates allow for no V servers. This however has the disadvantage (as in all

schemes involving depersonalization of votes only) that votes filled in ways that should not be allowed by the software of the Voting Application cannot be traced back to their origin. With **V** servers in place votes with invalid zero-knowledge proofs can be linked to the identity of the voter. Therefore there is a significant role to play for **V**-servers.

- The "Homomorphic Count" is a server where votes with valid zero-knowledge proofs are counted. Further, write-in votes and the electronic version of the information linking electronic and paper ballots can be taken in to do a full verification. In an interaction with a trusted group of people each holding a secret share of the private keys of the election, the result of the election is decrypted. A complete audit trail with zero-knowledge proofs that everything has been done correctly is produced and stored/exported for external audit.
- The **TS** servers are threshold servers, applications that allow the key share holders to use their key shares for decrypting the result of the election.
- The "External Audit Facility" is a facility that checks that the steps carried out by the **V** servers and the homomorphic count were performed correctly.

Please notice that not all relevant arrows are included in the drawing. For example arrows with origin at the key generation server have been left out for simplicity. Further, feed-back is helpful in a number of situations in order to deal with error and fraud situations. For example feed-back from the **V**-servers to the collection point is preferable in the case, where there are votes with valid content but invalid zero-knowledge proofs.

A special variant of a user interface to the local check program will also be claimed and an embodiment is described below:
- The container for collecting ballots is separated into two or more physical containers.
- When the voter wants to submit his vote he physically interacts with the device resulting in the device bringing itself in a mode, where it is possible for the voter to enter his ballot in at least one of the containers but not both/all.
- The ballots entered into one/some of the containers will be subjected to checks against the electronic ballots, possibly different types of checks depending on the container, whereas the ballots entered into entered into (the) other container(s) will not be checked against electronic ballots.

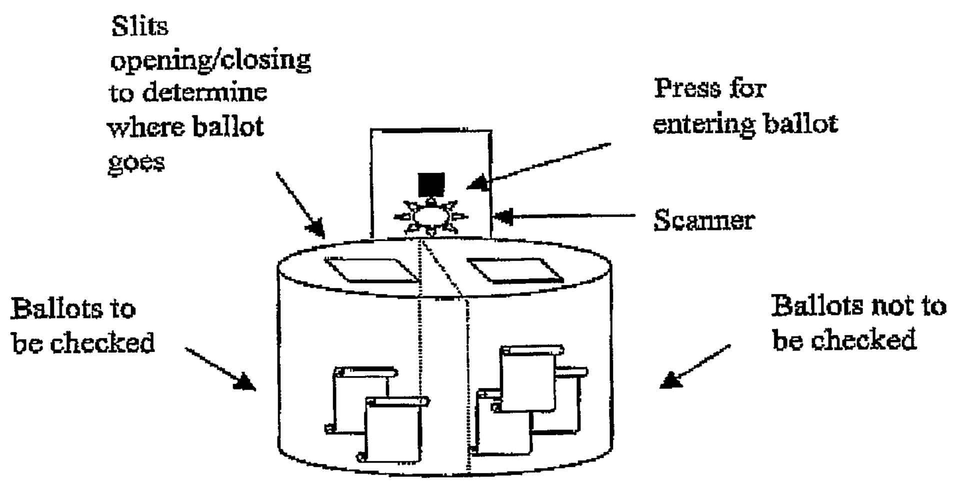

Slits opening/closing to determine where ballot goes
Press for entering ballot
Ballots to be checked
Ballots not to be checked

Device for selecting ballots to be checked

A more sophisticated version of such a device is also possible. In addition to a button to press for entering a ballot a scanner is available. The procedure is as follows:
- The voter presses the button (or in another way makes aware that the device must make its choice).
- The device indicates which slot will be opened, for example by lightening up the slot to be opened.
- The voter uses a scanning device to scan the information on his ballot linking it to an electronic ballot.
- The device opens the slot indicated.
- The voter enters his vote.

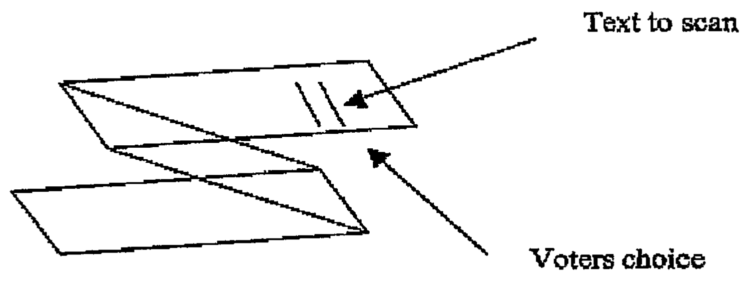
In this way the manual ballots will all be processed during the election with exception of the reading of the content of ballots to be checked. This simplifies the step after the election is over to actually enter the content of the ballots to be checked.

The two steps, first pressing the button, then scanning the ballot, are there to ensure that it will be impossible for the electronic voting system to signal to the device in a reliable way, which ballots shall not be checked.


Slits opening/closing to determine where ballot goes
Press for entering ballot
Scanner
Ballots to be checked
Ballots not to be checked

Device for selecting ballots to be checked with scanner

In order to apply this scheme it is preferable to form the ballots in a way such that the information linking them to electronic ballots can be scanned without revealing the content of the ballot. This can be done by having the information linking the physical and electronic ballots written on the back side of the physical ballots near the top or the bottom of the ballot.



Ballot with scannable text field.

Full Accountability

When the election is over a lot of options are available for verification and fine counting, providing full accountability of the system:

1) Verifying correctness of the verification and counting by an independent organisation using independent software. This is standard universal verifiability carried out at the "External Audit Facility".
2) Verifying the Signer log against the votes. In particular verifying that there is no systematic double signing. Voters who have signed more than once can be double-checked for, whether they got the permission (log from "Entry Election Application").
3) Selecting an adequate number of randomly chosen depersonalized votes for the whole country (a predetermined number, for example about 459 (or more) in our proposed solution). Do a test that each of those votes corresponds to a manual vote by a manual procedure in election districts.
4) For each district, selecting an adequate number of randomly chosen votes (a predetermined number, for example about 194 in our proposed solution). Do a test that each of those votes corresponds to a manual vote by a manual procedure in election districts. In contrary to 3), this work can be distributed over months (however, in the example embodiments given, it is done just after the election or even in parts during the election).

If 1)-4) are all successful and no other factors indicate that there is increased risk that this election has been tampered with, it will be natural to stop here. If however, one of the tests is not successful, a number of steps can be taken.

5) Electronic logs from voting sites can be compared to central logs from the Signer and the counting facilities. The result of this comparison may give an indication about, in which parts of the country a closer investigation shall take place.
6) Selected or all districts can perform a manual recount.
7) Selected or all districts can perform an extended manual recount involving the following: All electronic votes cast in the district are depersonalized in a MIX net. Each electronic vote is matched with a printed ballot.
8) In districts where the abovementioned pairing of electronic and manual votes can not be performed with sufficient success, a new election may be called for.
9) It is also possible with the help of highly trusted persons holding shares of the key used for decrypting the result of the election, to call in voters and have their votes decrypted such that they can judge about, whether fraud has taken place in the manual or the electronic system.

Conclusion

We observe that with the embodiments of the system proposed, benefits of several kinds can be achieved:

- Cost savings: For elections carried out in an orderly fashion, costs for counting can be limited significantly by having few locations, where counting takes place and counting votes almost 100% electronically.
- Increased services to voters: If the system is designed to do so, voting from arbitrary voting sites for each voter is possible because everything is electronic.
- Security: If the result of an election is disputed, there is much better accounting that in a manual election because the printed ballots can be compared to the electronic ones to establish which ballots have been tampered with.

Not all embodiments are optimal on each individual category, for example Internet-voting systems without security features build in optimize the first two while completely sacrificing the third. However, we describe a good compromise and leave a lot of room for election organisers to select just the solution that meets their requirements optimally. For example the scheme described is compatible with having Internet-voting also for selected categories of voters, like voters living abroad.

CLAIMS

What is claimed is:

1. A voting system feature comprising: at least one device used for voting entering preferably (the same or associated information) on a printed ballot and an encrypted electronic ballot linking the two to each other; preferably. Each voter will be allowed to watch the content of the paper ballot to verify that it contains his choices preferably. At least one instance making available depersonalised clear-text electronic ballots with their information linking them to printed ballots to the public or to selected entities; preferably. A procedure selecting a random sample of electronic ballots and verifying that their content correspond to the content of corresponding paper ballots with the purpose of establishing confidence that the electronic ballots have not been subjected to large-scale tampering.

2. A voting system feature comprising: at least one device used for voting entering preferably (the same or associated information) on a printed ballot and an encrypted electronic ballot linking the two to each other; preferably. Each voter will be allowed to watch the content of the paper ballot to verify that it contains his choices preferably. At least one instance making available depersonalised clear-text electronic ballots with their information linking them to printed ballots to the public or to selected entities; preferably. A procedure selecting a random sample of electronic ballots and verifying that their content correspond to the content of corresponding paper ballots with the purpose of establishing a deterrent against tampering with the voting device in individual election districts.

3. A device for collecting ballots comprising: two or more containers for collecting filled ballots and a user interface allowing a voter to make aware of his intention to submit his ballot arranged in such a way that it is decided at random at the time of ballot submission whether ballots shall be checked. This works in the way that it is by mechanical means ensured that ballots selected for checking at random are entered in a particular subset of containers.

4. A protocol for producing a zero-knowledge proof of a correctly performed combination of permuting and partial decryption of homomorphically encrypted messages and preferably the non-interactive versions of the protocol obtained by using the Fiat-Shamir heuristic.

5. A homomorphic commitment system that performs efficiently by making use of subgroups of Zn* for the message space and/or the randomization space.

6. A protocol comprising: use of a homomorphic verification system for verifying the correctness of the result of repeatedly permuting and re-encrypting and finally decrypting homomorphically encrypted content.

7. A protocol comprising: use of a homomorphic verification system for verifying the correctness of the write-in votes obtained by repeatedly permuting and re-encrypting and finally decrypting homomorphically encrypted votes.

8. A protocol comprising: use of a homomorphic verification system for verifying the correctness of the information linking electronic and printed ballots obtained by repeatedly permuting and re-encrypting and finally decrypting homomorphically encrypted votes.